Question 1:

Write a small explanation of all commands in Data Control Language (DCL) and Transaction Control Language (TCL), along with some examples.

ANS:

**Commands in Data Control Language (DCL) :**

1). Grant - This command gives the user access privileges to database

* Syntax : **GRANT SELECT, UPDATE ON MY\_TABLE TO USER1, USER2;**

**Example :**

GRANT SELECT, INSERT, SELECT, UPDATE ON Users TO ‘ Name’@’ localhost;

GRANT ALL ON Users TO ‘ Name’@’ localhost;

GRANT SELECT ON Users TO ‘ **\***’@’ localhost; (**\* Refers to all users in Database)**

2). Revoke - This command will withdraw the user’s access privileges given by using grant command.

* Syntax : **REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;**

**Example :**

REVOKE SELECT, INSERT, SELECT, UPDATE ON Users FROM ‘ Name’@’ localhost;

REVOKE ALL ON Users FROM ‘ Name’@’ localhost;

REVOKE SELECT ON Users FROM ‘ **\***’@’ localhost;

**Commands in Transaction Control Language (TCL) :**

TCL is used to manage the transaction in relational database / used to manage transactions within the database.

1. Commit 2. RollBack 3. SavePoint

1). Commit :

The COMMIT command in SQL is used to permanently save any transaction into the database. Generally, whenever we use any DML command such as INSERT, UPDATE, or DELETE, the changes made by these commands are not permanent. Hence, before closing the current session, we may roll back any changes made through these commands.

Due to the above reason, it is mandatory to use the COMMIT command to mark any changes made as permanent.

Note : If auto commit is enabled then all the transaction will automatically be saved in to data base.

🡺 Once we have done a **COMMIT**, we cannot undo it. We can, however, rollback, but doing a rollback will undo the entire transaction.

* Syntax **: COMMIT**;
* We must write the COMMIT command before and after every DDL command to save the change permanently in the database.
* Once COMMIT is performed, it cannot be undone unless it is rolled back.

2). RollBack :

The rollback command in TCL is used for **restoring the database to the last committed state**. In other words, the rollback command restores the database to its original state since the last command that was committed.

The rollback command will basically revert or roll back any changes that were not committed during our transaction using the COMMIT command.

* Syntax : ROLLBACK;
* ROLLBACK is used to undo the transactions that have not already been permanently saved (or committed) to the database.
* The ROLLBACK command restores the previously stored value, that is, the data present before the execution of the transactions.

Interestingly, the rollback command can also be used with the SAVEPOINT command to jump to a savepoint in any ongoing transaction.

* Syntax : SAVEPOINT savepoint\_name;

EX: if we already given Savepoint A;

Savepoint B;

Savepoint C;

Then to roll back 🡺 **Rollback to B;** then the transaction after savepont B will not be stored and we will not get savepoint C transaction in output.

3). Savepoint :

The SAVEPOINT command in TCL is basically used to temporarily save a transaction so that we can roll back to that point (saved point) whenever required.( used for Rollback purpose)

* Syntax : SAVEPOINT savepoint\_name;

Also, we can release a savepoint, meaning, remove any particular savepoint. The syntax to release a savepoint is given below :

* Syntax : RELEASE SAVEPOINT savepoint\_name;

Question 2:

create an table "customer" with following columns

cust\_id -- small int

cust\_no -- medium int

cust\_firstname -- char

cust\_lastname -- varchar

cust\_gender -- varchar

cust\_dob -- DATE

cust\_age -- int

cust\_revenue -- decimal

cust\_spend\_amt -- float

cust\_address -- TEXT

cust\_city -- CHAR

cust\_comments -- TEXT

cust\_status -- ENUM (active, Inactive)

cust\_available -- BOOLEAN

Note : column size as per your wish and enter 5 records into the table

**CODE:**

Create table Customer (

cust\_id Smallint,

cust\_no mediumint,

cust\_firstname char(20),

cust\_lastname varchar(20),

cust\_gender varchar(10),

cust\_dob DATE,

cust\_age int,

cust\_revenue Decimal,

cust\_spend\_amt Float,

cust\_address TEXT,

cust\_city CHAR(20),

cust\_comments TEXT,

cust\_status ENUM('active', 'Inactive'),

cust\_available BOOLEAN

);

select\*from customer;

INSERT INTO Customer (cust\_id, cust\_no, cust\_firstname, cust\_lastname, cust\_gender, cust\_dob, cust\_age, cust\_revenue, cust\_spend\_amt, cust\_address, cust\_city, cust\_comments, cust\_status, cust\_available)

VALUES

(1, 1001, 'John', 'Doe', 'Male', '1990-05-15', 31, 50000.00, 1500.00, '123 Main St', 'New York', 'Regular customer', 'active', true),

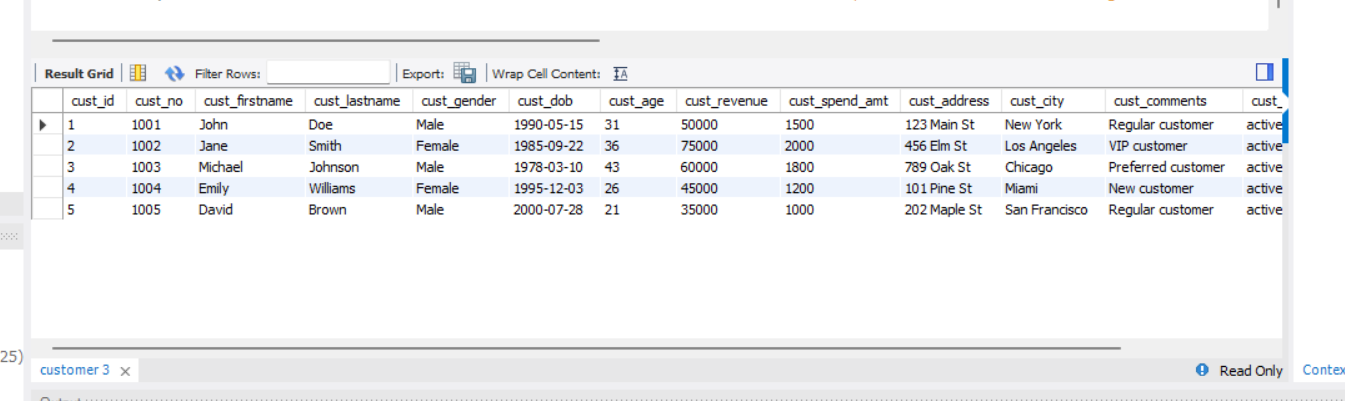
(2, 1002, 'Jane', 'Smith', 'Female', '1985-09-22', 36, 75000.00, 2000.00, '456 Elm St', 'Los Angeles', 'VIP customer', 'active', true),

(3, 1003, 'Michael', 'Johnson', 'Male', '1978-03-10', 43, 60000.00, 1800.00, '789 Oak St', 'Chicago', 'Preferred customer', 'active', true),

(4, 1004, 'Emily', 'Williams', 'Female', '1995-12-03', 26, 45000.00, 1200.00, '101 Pine St', 'Miami', 'New customer', 'active', true),

(5, 1005, 'David', 'Brown', 'Male', '2000-07-28', 21, 35000.00, 1000.00, '202 Maple St', 'San Francisco', 'Regular customer', 'active', true);

**OUTPUT:**



Question 3:

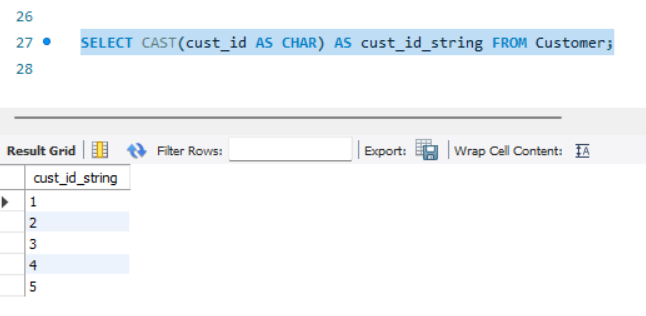
The output of the SELECT statement should be a single column

1. convert a value of the cust\_id column to string.

**CODE:**

SELECT CAST(cust\_id AS CHAR) AS cust\_id\_string FROM Customer;

**OUTPUT**:

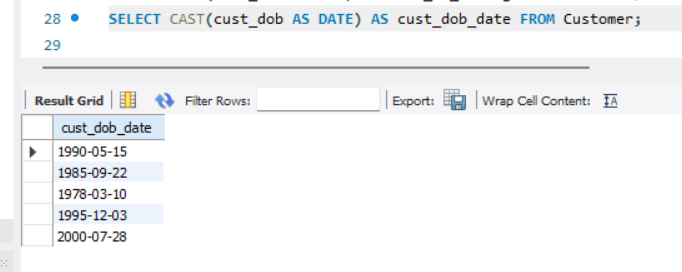


1. convert a value of the cust\_dob column to date.

**CODE:**

SELECT CAST(cust\_dob AS DATE) AS cust\_dob\_date FROM Customer;

**OUTPUT**:

****

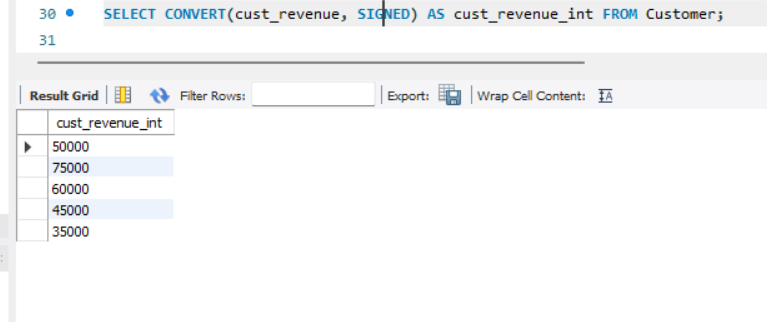
1. convert a value of the cust\_revenue column to int

**CODE:**

Note : We cannot directly cast a DECIMAL column to INT using CAST. So that we can use CONVERT.

SELECT CONVERT(cust\_revenue, SIGNED) AS cust\_revenue\_int FROM Customer;

**OUTPUT**:

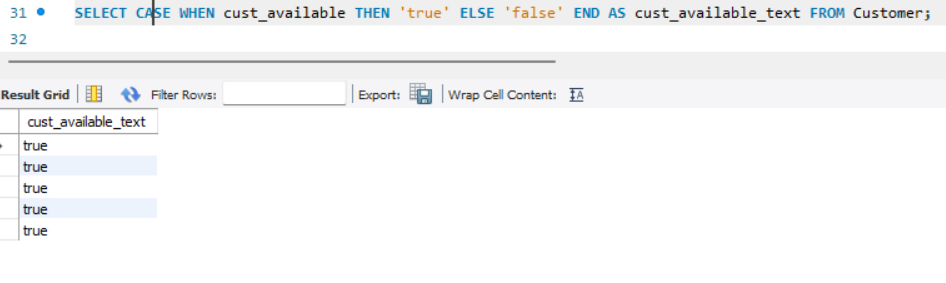


4) convert a value of the cust\_available column to text

**CODE**:

SELECT CASE WHEN cust\_available THEN 'true' ELSE 'false' END AS cust\_available\_text FROM Customer;

**OUTPUT**:



Question 4:

Return "Child" if the customer's age is less than 18,

“Adult Customer” if the customer's age is between 21 and 65,

and "Senior Customer" if the customer's age is greater than 65.

**CODE**:

SELECT CASE

WHEN cust\_age < 18 THEN 'Child'

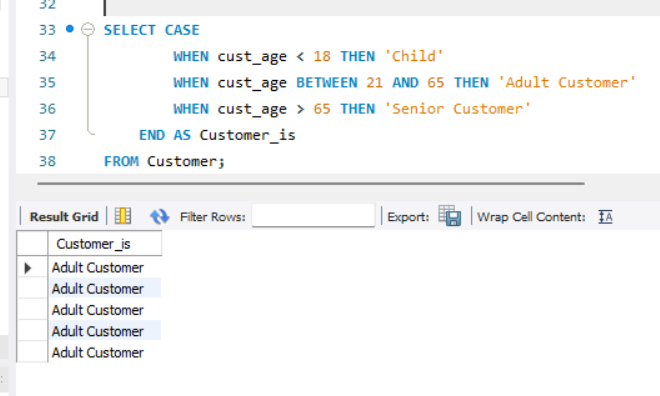
WHEN cust\_age BETWEEN 21 AND 65 THEN 'Adult Customer'

WHEN cust\_age > 65 THEN 'Senior Customer'

END AS Customer\_is

FROM Customer;

**OUTPUT:**



Question 5:

Alter the above table by adding the following constraints

cust\_id -- not null

cust\_no -- unique

cust\_city -- default with a value NY

**CODE:**

ALTER TABLE Customer MODIFY COLUMN cust\_id SMALLINT NOT NULL;

ALTER TABLE Customer ADD CONSTRAINT UNIQUE (cust\_no);

ALTER TABLE Customer MODIFY COLUMN cust\_city CHAR(20) DEFAULT 'NY';

Question 6:

Create a table called 'Employees' with the following columns:

EmployeeID as an integer primary key

Name as a non-null varchar(50)

Age as an integer with a check constraint that ensures the age is greater than or equal to 18

Department as a varchar(50) with a default value of 'IT'.

Insert some value and if its failed to insert, will provide the some comment on why it was failing

**CODE:**

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Age INT CHECK (Age >= 18),

Department VARCHAR(50) DEFAULT 'IT'

);

INSERT INTO Employees (EmployeeID, Name, Age, Department) VALUES (1, 'John', 25, 'HR');

INSERT INTO Employees (EmployeeID, Name, Age) VALUES (2, 'Jane', 17, 'Marketing'); // This insert is failing due to age constraint that we set

INSERT INTO Employees (EmployeeID, Name, Age, Department) VALUES (3, 'Johnson', 30, 'Finance');

INSERT INTO Employees (EmployeeID, Name, Age, Department) VALUES (4, 'William', 22, 'IT');

INSERT INTO Employees (EmployeeID, Name, Age, Department) VALUES (5, 'David', 16); // This insert is failing due to age constraint that we set

Question 7:

Create an table "customers\_data" with below columns

customer\_id -- INT -- PRIMARY KEY

customer\_name -- VARCHAR(255) -- NOT NULL,

Create an table "orders" with below columns

order\_id -- INT -- PRIMARY KEY ,

customer\_id -- INT -- NOT NULL ,

order\_date -- DATE

Make an connection(link) between customers\_data and orders tables

explain what operation you did for that.

**CODE:**

CREATE TABLE customers\_data (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(255) NOT NULL

);

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

customer\_id INT NOT NULL,

order\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers\_data(customer\_id)

);

🡺In “customers\_data” table customer\_id is PRIMARY KEY which REFENCES in “Orders” table. We can link this both table by using FOREIGN key of Customer\_id from “Order table”

Question 8:

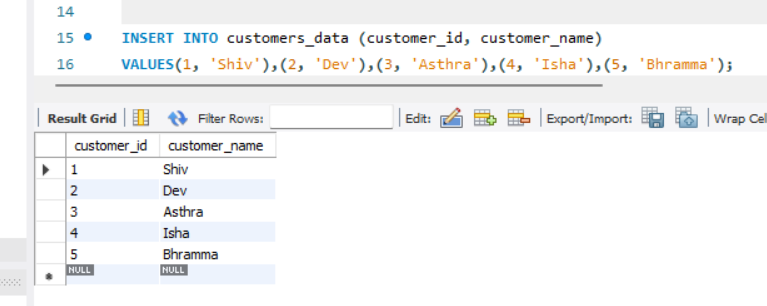
Enter some random 5 records of data into customers\_data and orders table.

**CODE:** (customers\_data)

INSERT INTO customers\_data (customer\_id, customer\_name)

VALUES (1, 'Shiv'),(2, 'Dev'),(3, 'Asthra'),(4, 'Isha'),(5, 'Bhramma');

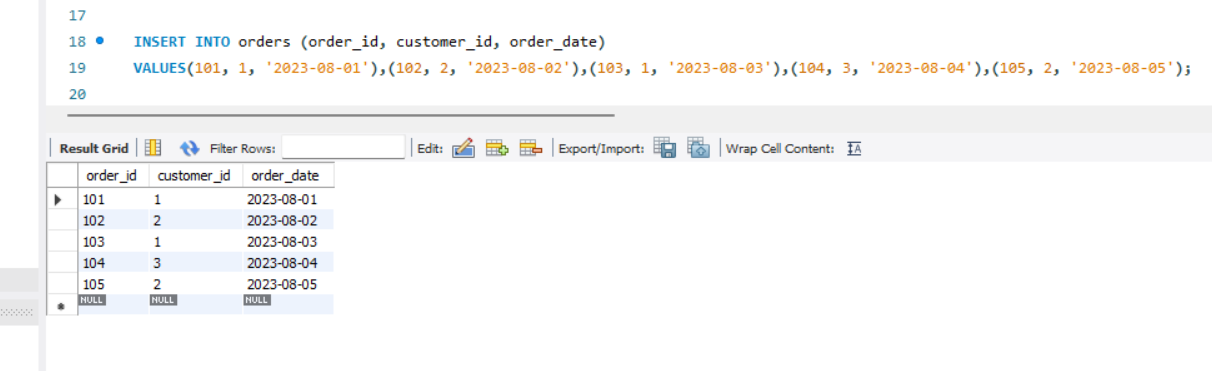
**OUTPUT**: (customers\_data)



**CODE:** (Orders table)

INSERT INTO orders (order\_id, customer\_id, order\_date)

VALUES (101, 1, '2023-08-01'),(102, 2, '2023-08-02'),(103, 1, '2023-08-03'),(104, 3, '2023-08-04'),(105, 2, '2023-08-05');

**OUTPUT**: (Orders table) 

Question 9:

Remove any of 2 records from the customers\_data and orders table.

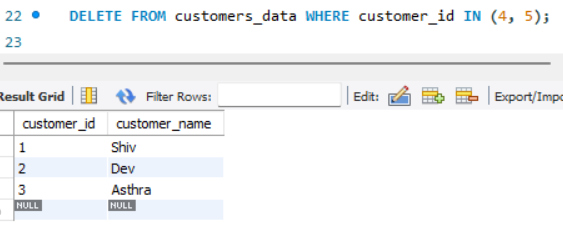
Explain how to remove the data from the two tables in a steps

🡺We can remove data by using DELETE command.

CODE(Customers\_data) :

DELETE FROM customers\_data WHERE customer\_id IN (4, 5);

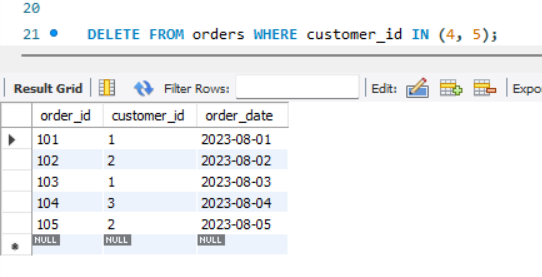
**OUTPUT**:



CODE (Order table):

DELETE FROM orders WHERE customer\_id IN (4, 5);

**OUTPUT**:



Question 10:

Difference between unique and primary key

🡺 The primary key will not accept NULL values whereas the Unique key can accept NULL values.

🡺 A table can have only one primary key whereas there can be multiple unique keys on a table.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **PRIMARY KEY** | **UNIQUE KEY** |
| **Basic** | Used to serve as a unique identifier for each row in a table. | Uniquely determines a row that isn’t the primary key. |
| **NULL value acceptance** | Cannot accept NULL values. | Can accept NULL values. |
| **Number of keys that can be defined in the table** | Only one primary key | More than one unique key |
| **Index** | Creates clustered index | Creates non-clustered index |
| **Auto Increment** | A Primary key supports auto-increment value. | A unique key does not support auto-increment value. |
| **Modification** | We cannot change or delete values stored in primary keys. | We can change unique key values. |
| **Uses** | The primary Key is used for indicating the rows uniquely. | The Unique Key is used for preventing duplicate entries. |
| **Syntax** | CREATE TABLE Student  (  Student\_Id INT PRIMARY KEY,  Student\_name VARCHAR(150),  roll\_number INT(10)  ) | CREATE TABLE House  (  House\_Number INT UNIQUE,  House\_Name VARCHAR(150),  House\_Address VARCHAR(250)  ) |

Differnce between float, double and decimal

FLOAT 🡺Stores approximate values(0-24).

DOUBLE🡺 Stores more than precise values than FLOAT(>25 bits)

DECIMAL🡺Stores the exact values.